Brain subtypes

Gabriel Altschuler

September 21, 2011

To validate the optimum threshold selected based on the tissue-specific data we repeated the cross-validation and precision-recall analysis using an independent, cross-species dataset to classify brain regions in an independent dataset of human, mouse and rat arrays.

1 Data

The required data and metadata is contained within the R package `pathprint` and in the shared directory on `hpc111`. We will use the `pathprint.v.0.3.beta4` build in this session. First we need to source the pathprint package and load the data libraries containing the fingerprint and the metadata. We will also load the tissue-specific data from a local file from the shared directory on `hpc111`.

```
> library(GMAfunctions)
> library(pathprint)
> load("/data/shared/Fingerprint/curatedCellTypes/brain.data.RData")
> #N<-c(seq(0.001, 0.01, 0.001), seq(0.02, 0.1,0.01), seq(0.2, 0.5, 0.1), seq(1, 7, 1))
> N<-c(0.001, seq(0.0025, 0.01, 0.0025), seq(0.02, 0.1,0.01), seq(0.2, 0.5, 0.1), 0.75,
```

2 Brain sub-types

We will define a set of Brain types that we are interesting in using as classifiers. We will select sub-types that are represented by at least 5 arrays in each species. These will be used to construct a PCA to assess to what extent the different brain subtypes are distinguished relative to the species separation.

```
> brainTypes<-c("Caudate Nucleus",
                "Cerebellum",
                "Frontal cortex",
                "Substantia nigra",
                "Hypothalamus",
                "Pituitary",
                "Amygdala",
                "Hippocampus",
```

```
                "Putamen",
                "Pineal")
> brain.data$Type<-NA
> for (i in brainTypes){
   ID<-grep(i, apply(
                brain.data[, c("Title", "Source", "Characteristics")],
                1, function(y){
                          paste(y, collapse = " ")
                          }
                     ), ignore.case = TRUE)

   if(sum(!(is.na(brain.data$Type[ID]))) > 0) print(
     list(a = paste("double assignment for tissue", i, "at "),
          b = brain.data[ID, "GSM"][!(is.na(brain.data$Type[ID]))]))
   brain.data$Type[ID][!is.na(brain.data$Type[ID])] <- NA
   brain.data$Type[ID][is.na(brain.data$Type[ID])] <- i
   }
$a
[1] "double assignment for tissue Hippocampus at "

$b
[1] "GSM53302" "GSM53303" "GSM53304" "GSM53308" "GSM53309"
[6] "GSM53310"
> brainTypes.table<-as.data.frame(table(brain.data$Type), stringsAsFactors = FALSE)
> colnames(brainTypes.table)<-c("Type", "nTotal")
> brainTypes.table$nMouse<-sapply(brainTypes.table$Type,
       function(x){sum(grepl("Mus musculus", subset(brain.data,
                                                    Type == x)$Species))})
> brainTypes.table$nRat<-sapply(brainTypes.table$Type,
       function(x){sum(grepl("Rattus norvegicus", subset(brain.data,
                                                    Type == x)$Species))})
> brainTypes.table$nHuman<-sapply(brainTypes.table$Type,
       function(x){sum(grepl("Homo sapiens", subset(brain.data,
                                                    Type == x)$Species))})
```

3 Principal Component Analysis

We can plot a PCA of the data from to provide a view on the inter vs intra sub-type spread.

```
> #brainTypes.sub<-brainTypes.table$Type[
> #  apply(brainTypes.table[,c("nMouse", "nHuman", "nRat")],
> #        1, function(x){sum(x < 5) == 0})
> #  ]
> brainTypes.sub<-brainTypes.table$Type[
```

	Type	nTotal	nMouse	nRat	nHuman
1	Amygdala	168	20	135	13
2	Caudate Nucleus	77	0	0	77
3	Cerebellum	147	50	2	95
4	Frontal cortex	134	41	0	93
5	Hippocampus	387	98	274	15
6	Hypothalamus	66	20	13	33
7	Pineal	26	0	26	0
8	Pituitary	128	6	99	23
9	Putamen	9	0	0	9
10	Substantia nigra	25	0	0	25

Table 1: Brain types and curated arrays

```
   apply(brainTypes.table[,c("nMouse", "nHuman", "nRat")],
         1, function(x){sum(x < 5) <= 1})
   ]
> # If desired can specify species or types
> # species.sub<-c("Homo sapiens", "Mus musculus", "Rattus norvegicus")
> # brainTypes.sub<-c("pituitary", "amygdala", "Frontal Cortex")
> brain.data.sub<-brain.data[brain.data$Type %in% brainTypes.sub,]
> # brain.data.sub<-brain.data.sub[brain.data.sub$Species %in% species.sub,]
>
> # load fingerprint data
> #data(GEO.fingerprint.matrix)
> #brain.fingerprints.sub<-GEO.fingerprint.matrix[,
> #    match(brain.data.sub$GSM, colnames(GEO.fingerprint.matrix))]
> # need to remove NA rows
> load(
  "/home/galtschu2/Documents/Projects/Fingerprinting/data/sq_.POE.matrix.2011-07-27.RDat
   )
> brain.POE.matrix<-POE.matrix[,
     match(brain.data.sub$GSM, colnames(POE.matrix))]
> na.match<-which(is.na(brain.POE.matrix), arr.ind = TRUE)
> brain.POE.matrix.naRm<-brain.POE.matrix[
   -unique(na.match[,"row"]),]
> brain.pc<-prcomp(t(brain.POE.matrix.naRm))
> colors<-rainbow((length(brainTypes.sub)))[
   as.factor(brain.data.sub$Type)]
> species<-(1:3)[as.factor(brain.data.sub$Species)]
> platforms<-(1:length(levels(as.factor(brain.data.sub$GPL))))[
   as.factor(brain.data.sub$GPL)]
```

```
> plot(brain.pc$x, col = colors, pch = platforms)
> legend("bottomleft", levels(as.factor(brain.data.sub$Type)),
       text.col = rainbow((length(brainTypes.sub)))[1:length(brainTypes.sub)],
        cex = 0.75, bty = "n")
> legend("bottomright",legend = levels(as.factor(brain.data.sub$GPL)),
       pch = (1:length(levels(as.factor(brain.data.sub$GPL)))),
        cex = 0.75, bty = "n")


> plot(brain.pc$x[,3], brain.pc$x[,4], col = colors, pch = platforms)
> legend("bottomleft", levels(as.factor(brain.data.sub$Type)),
       text.col = rainbow((length(brainTypes.sub)))[1:length(brainTypes.sub)],
        cex = 0.75, bty = "n")
> legend("bottomright",legend = levels(as.factor(brain.data.sub$GPL)),
       pch = (1:length(levels(as.factor(brain.data.sub$GPL)))),
        cex = 0.75, bty = "n")
```

	Total	Mouse	Rat	Human
Amygdala	168	20	135	13
Cerebellum	147	50	2	95
Frontal cortex	134	41	0	93
Hippocampus	387	98	274	15
Hypothalamus	66	20	13	33
Pituitary	128	6	99	23

Table 2: Brain types used in threshold optimization analysis

4 Classification by nearest neighbor

We will now use a simple nearest-neighbor approach to identify the closest human array to each mouse array. This gives us the predicted tissue. We will cycle through a range of threshold values. For expediency this can be run this in parallel on the server using the package `muilticore`. This will be performed on thresholded values of the fingeprint The next step is to define a function to convert a POE matrix to a ternary matrix, according to a threshold.

```
> ternaryThreshold <- function(matrix, threshold)
   # function to convert a POE matrix to a thresholded matrix
   {
   high<-threshold
   low<-(-threshold)
   threshold.matrix<-(matrix>high)-(matrix<low)
   threshold.matrix[is.na(threshold.matrix)]<-0
   return(threshold.matrix)
   }


> library(multicore)
> library(doMC)
> # run over 15 cores - i.e. 2 thresholds per core
> registerDoMC(cores = 15)
> err.threshold<-function(source.data, matching.data, source.matrix, matching.matrix)
 {
   foreach (j = 1:30) %dopar% {
   source.data$predictedTissue<-NA
   n <- N[j]
   threshold<-10^(-n)
   matching.threshold<-ternaryThreshold(matching.matrix, threshold)
   source.threshold<-ternaryThreshold(source.matrix, threshold)
   ordered<-vector("list", nrow(source.data))
   for (i in 1:nrow(source.data)){
     source.fingerprint<-source.threshold[,source.data$GSM[i], drop = FALSE]
     ordered <- sort(colSums(abs(apply(
```

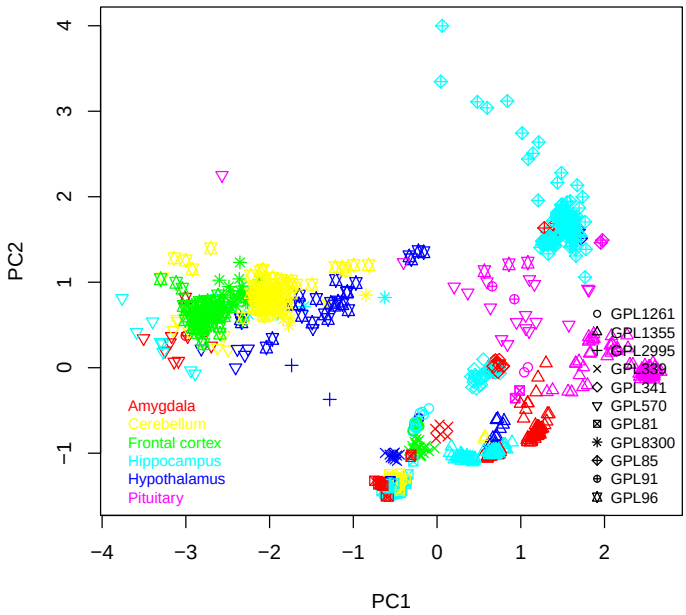

Figure 1: Plots of the two most significant principal components for brain sub-type gene expression data processed using POE values

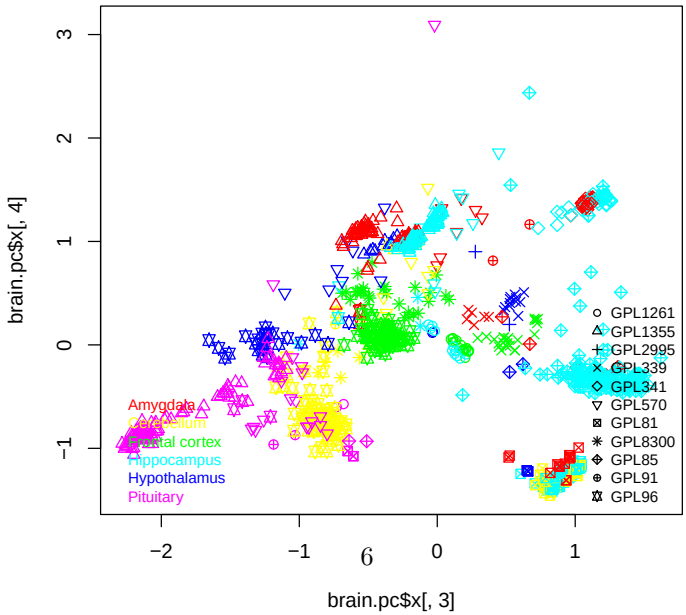

Figure 2: Plots of the thrid and fourth most significant principal components for brain sub-type gene expression data processed using POE values

```
                        matching.threshold, 2, function(x){x - source.fingerprint}
                                            ))))
      source.data$predictedTissue[i]<-matching.data$Type[
                                      matching.data$GSM == names(ordered)[1]
                                        ]
      }
     Actual <- as.factor(source.data$Type)
     Predicted<-factor(source.data$predictedTissue, levels = levels(Actual))
     m <- table(Actual = Actual, Predicted = Predicted)
     err <- c(threshold, 1 - sum(diag(m)) / sum(m))
     return(list(err = err, m = m))
     }
     }
> human.data<-subset(brain.data.sub, Species == "Homo sapiens")
> human.matrix<-brain.POE.matrix.naRm[,
     match(human.data$GSM, colnames(brain.POE.matrix.naRm))]
> mouse.data<-subset(brain.data.sub, Species == "Mus musculus")
> mouse.matrix<-brain.POE.matrix.naRm[,
     match(mouse.data$GSM, colnames(brain.POE.matrix.naRm))]
> rat.data<-subset(brain.data.sub, Species == "Rattus norvegicus")
> rat.matrix<-brain.POE.matrix.naRm[,
     match(rat.data$GSM, colnames(brain.POE.matrix.naRm))]
> human.mouse.class<-err.threshold(mouse.data, human.data, mouse.matrix, human.matrix)
> human.mouse.class.m<-lapply(human.mouse.class, function(x){x[["m"]]})
> human.mouse.class.err<-t(sapply(human.mouse.class, function(x){x[["err"]]}))
> human.rat.class<-err.threshold(rat.data, human.data, rat.matrix, human.matrix)
> human.rat.class.m<-lapply(human.rat.class, function(x){x[["m"]]})
> human.rat.class.err<-t(sapply(human.rat.class, function(x){x[["err"]]}))
> rat.mouse.class<-err.threshold(mouse.data, rat.data, mouse.matrix, rat.matrix)
> rat.mouse.class.m<-lapply(rat.mouse.class, function(x){x[["m"]]})
> rat.mouse.class.err<-t(sapply(rat.mouse.class, function(x){x[["err"]]}))
> mouse.human.class<-err.threshold(human.data, mouse.data, human.matrix, mouse.matrix)
> mouse.human.class.m<-lapply(mouse.human.class, function(x){x[["m"]]})
> mouse.human.class.err<-t(sapply(mouse.human.class, function(x){x[["err"]]}))
> rat.human.class<-err.threshold(human.data, rat.data, human.matrix, rat.matrix)
> rat.human.class.m<-lapply(rat.human.class, function(x){x[["m"]]})
> rat.human.class.err<-t(sapply(rat.human.class, function(x){x[["err"]]}))
> mouse.rat.class<-err.threshold(rat.data, mouse.data, rat.matrix, mouse.matrix)
> mouse.rat.class.m<-lapply(mouse.rat.class, function(x){x[["m"]]})
> mouse.rat.class.err<-t(sapply(mouse.rat.class, function(x){x[["err"]]}))
```

The confusion matrix is used to calculate the error rate, as shown for a threshold of 0.001.

```
> human.mouse.class.m[[12]]
                Predicted
Actual           Amygdala Cerebellum Frontal cortex
```

```
  Amygdala             0          6             0
  Cerebellum           0         40             0
  Frontal cortex       0         41             0
  Hippocampus          0         45             0
  Hypothalamus         0         18             1
  Pituitary            0          0             0
               Predicted
Actual          Hippocampus Hypothalamus Pituitary
  Amygdala                8            0         6
  Cerebellum             10            0         0
  Frontal cortex          0            0         0
  Hippocampus            53            0         0
  Hypothalamus            0            1         0
  Pituitary               0            0         6
```

This table gives an error rate of

```
> 1 - sum(diag(human.mouse.class.m[[12]])) / sum(human.mouse.class.m[[12]])
[1] 0.5744681
```

This can be repeated for the other combinations. N.B. this code is hidden for brevity

```
> par(mfcol = c(2,3), cex.main = 1)
> plot(human.mouse.class.err, log = "x", xlab = "Ternary threshold", ylab = "Error rate"
     main = "Nearest neighbor, mouse to human", ylim = c(0,1))
> plot(human.rat.class.err, log = "x", xlab = "Ternary threshold", ylab = "Error rate",
     main = "Nearest neighbor, rat to human", ylim = c(0,1))
> plot(rat.mouse.class.err, log = "x", xlab = "Ternary threshold", ylab = "Error rate",
     main = "Nearest neighbor, mouse to rat", ylim = c(0,1))
> plot(rat.human.class.err, log = "x", xlab = "Ternary threshold", ylab = "Error rate",
     main = "Nearest neighbor, human to rat", ylim = c(0,1))
> plot(mouse.human.class.err, log = "x", xlab = "Ternary threshold", ylab = "Error rate"
     main = "Nearest neighbor, human to mouse", ylim = c(0,1))
> plot(mouse.rat.class.err, log = "x", xlab = "Ternary threshold", ylab = "Error rate",
     main = "Nearest neighbor, rat to mouse", ylim = c(0,1))
```

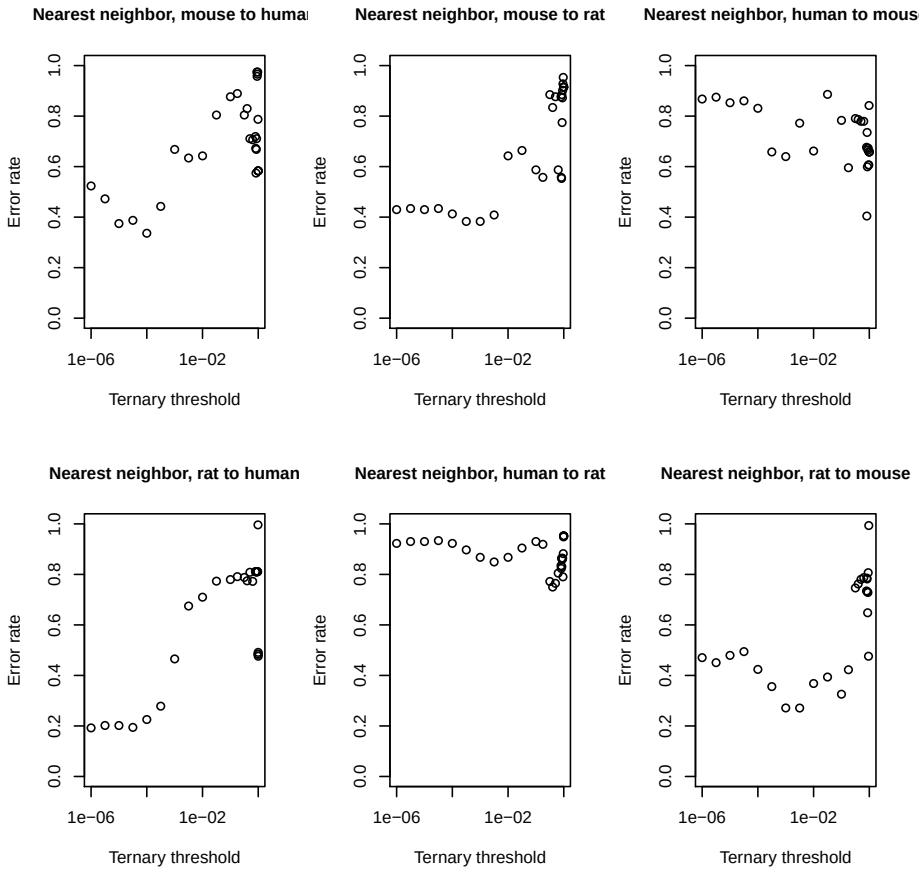

Figure 3: Error rate vs fingerprint threshold

5 Precision-Recall

Here we will group all samples and construct PR curves for each one, irrespective of it's source. This will test the ability of the fingerprint to classify samples both within and between platforms

```
> class<-brain.data.sub$Type[
     match(colnames(brain.POE.matrix.naRm), brain.data.sub$GSM)]
> names(class)<-colnames(brain.POE.matrix.naRm)
> PR.samples.all<-invisible(foreach (j = 1:30) %dopar% {
   n <- N[j]
   threshold<-10^(-n)
   full.threshold<-ternaryThreshold(brain.POE.matrix.naRm, threshold)
   pr.sample.all<-matrix(nrow = 100, ncol = ncol(full.threshold))
   av.precision<-vector("numeric", ncol(full.threshold))
   colnames(pr.sample.all)<-colnames(full.threshold)
   for (i in 1:ncol(full.threshold)){
      PR <- precisionRecall(
       testMatrix = full.threshold[,i, drop = FALSE],
       recallMatrix = full.threshold[,-i],
       testClass = class[i],
       recallClass = class[-i]
       )
     pr.sample.all[,i]<-approx(
                 x = PR$recall,
                 y = PR$precision,
                 xout = seq(0.01, 1, 0.01),
                 rule = 2
                 )$y
     av.precision[i]<-attr(PR, "Av.precision")
     }
   interpol <- list(
     PR = rowMeans(pr.sample.all),
     MAP = mean(av.precision)
     )
   })


> PR.samples.unthresholded.all<-invisible(foreach (i = 1:ncol(brain.POE.matrix.naRm)) %d
  PR <- precisionRecall(
   testMatrix = brain.POE.matrix.naRm[,i, drop = FALSE],
   recallMatrix = brain.POE.matrix.naRm[,-i],
   testClass = class[i],
   recallClass = class[-i],
   method = "euclidean"
   )
   PR.samples.unthresholded<-approx(
```

```
                x = PR$recall,
                y = PR$precision,
                xout = seq(0.01, 1, 0.01),
                rule = 2
                )$y

    interpol<-list(
      PR = PR.samples.unthresholded,
      AP = attr(PR, "Av.precision")
      )
  })
> unthresholded.MAP<-mean(unlist(
  lapply(PR.samples.unthresholded.all, function(x){x[["AP"]]}))
                  )
> PR.samples.unthresholded.all<-rowMeans(as.data.frame(
  lapply(PR.samples.unthresholded.all, function(x){x[["PR"]]})
  ))
```

6 Random genesets

The fingerprint was establish based on the hypothesis that aggregating genes based on pathways provides a means to integrate 'expert' prior biological knowledge with gene expression data. To test that this knownledge contributes to the success of the fingerprint we have constructed a 'random' fingerprint based on gene sets produced by randomly sampling the pathway fingeprint gene sets. The size distribution has been retained, as has the overall list and frequency of gene IDs.

```
> load("~/Documents/Projects/Fingerprinting/data/random_sq_.POE.matrix.2011-08-08.RData"
> brain.POE.matrix.random<-POE.matrix[,
    match(brain.data.sub$GSM, colnames(POE.matrix))]
> na.match.random<-which(is.na(brain.POE.matrix.random), arr.ind = TRUE)
> if(dim(na.match.random)[1] > 0){
 brain.POE.matrix.naRm.random<-brain.POE.matrix.random[
  -unique(na.match.random[,"row"]),]
  } else {
    brain.POE.matrix.naRm.random <- brain.POE.matrix.random
  }
> class.random<-brain.data.sub$Type[
    match(colnames(brain.POE.matrix.naRm.random), brain.data.sub$GSM)]
> names(class.random)<-colnames(brain.POE.matrix.naRm.random)
> PR.samples.all.random<-invisible(foreach (j = 1:30) %dopar% {
  n <- N[j]
  threshold<-10^(-n)
  full.threshold<-ternaryThreshold(brain.POE.matrix.naRm.random, threshold)
```

```
> op<-par(mfrow = c(1,1), pty = "s")
> par(mfrow = c(5,6), mar = c(1,1,1,1))
> n <- N
> for (i in 1:30){
   plot.new()
   axis(1, seq(0,1,0.5))
   axis(2, seq(0,1,0.5))
   title(paste("10^", n[i], sep = ""))
   lines(x = seq(0.01, 1, 0.01), y = PR.samples.all[[i]]$PR)
   }
>   par(op)
```

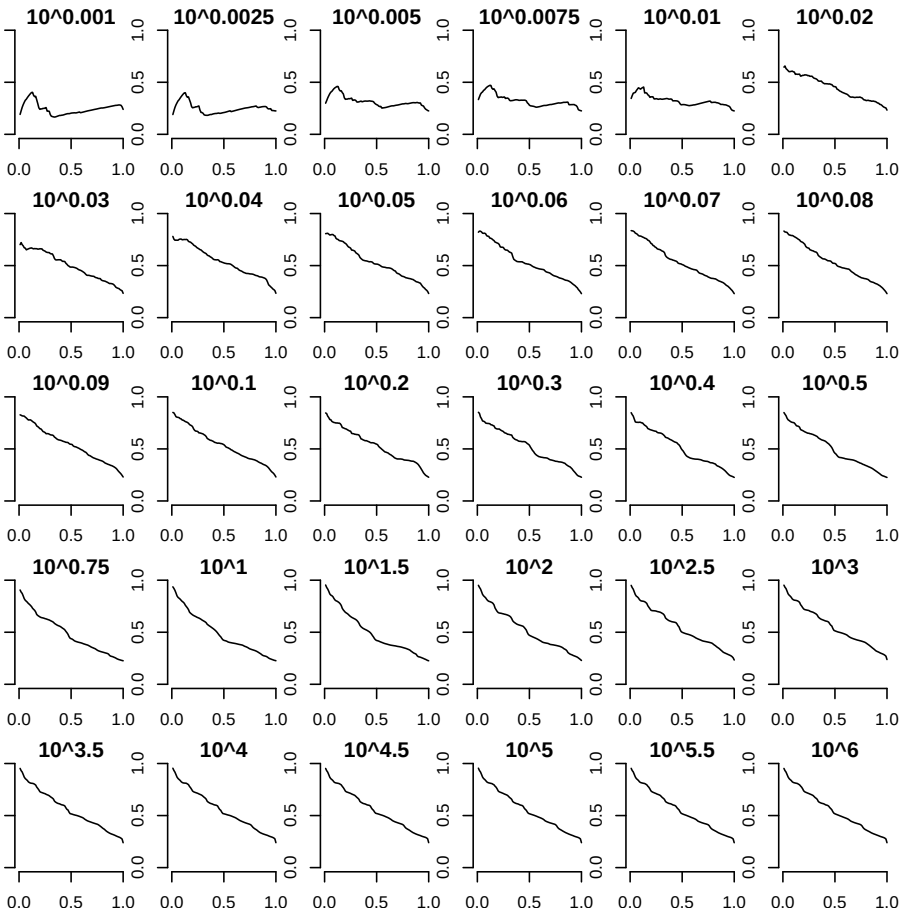

Figure 4: Precision-recall curves for aggregated data

```
> plot(x = threshold<-10^(-N),
       y = unlist(lapply(
         PR.samples.all, function(x){x[["MAP"]]}
         )),
       log = "x",
       xlab = "Ternary threshold",
       ylab = "Mean average precision",
       ylim = c(0,1)
       )
> abline(h = unthresholded.MAP, col = "red")
```

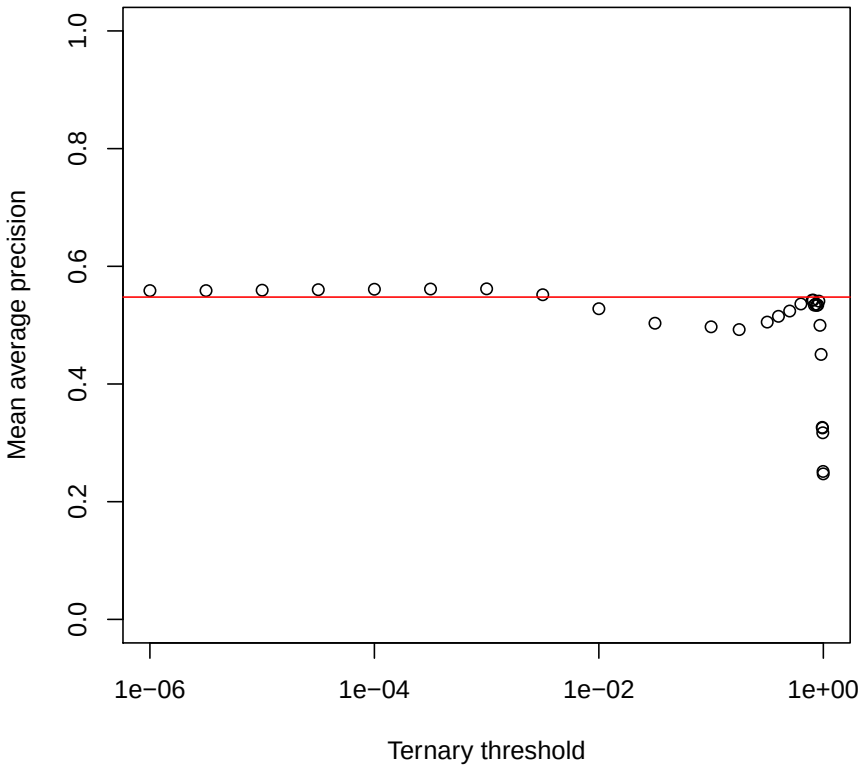

Figure 5: Plot of the mean average precision over the range of thresholds tested for the aggregated dataset (approach 3). Solid lines indicates the mean average precision for the unthresholded fingerprint (red)

```
pr.sample.all<-matrix(nrow = 100, ncol = ncol(full.threshold))
av.precision<-vector("numeric", ncol(full.threshold))
colnames(pr.sample.all)<-colnames(full.threshold)
for (i in 1:ncol(full.threshold)){
   PR <- precisionRecall(
    testMatrix = full.threshold[,i, drop = FALSE],
    recallMatrix = full.threshold[,-i],
    testClass = class.random[i],
    recallClass = class.random[-i]
    )
  pr.sample.all[,i]<-approx(
              x = PR$recall,
              y = PR$precision,
              xout = seq(0.01, 1, 0.01),
              rule = 2
              )$y
  av.precision[i]<-attr(PR, "Av.precision")
  }
interpol <- list(
  PR = rowMeans(pr.sample.all),
  MAP = mean(av.precision)
  )
})
```

7 Intra- vs inter-class distance

The ratio of the within-class to between-class variance can be used as another means to quantify the extent to which the fingerprints formed discrete brain types. Variance was defined as sum of the squared distance (Euclidean or Manhattan) between each sample and the mean pathway fingerprint for each brain type.

```
> varRatio<-function(matrix, colClasses, method = "manhattan"){
   # check values
   if (!(all.equal(colnames(matrix), names(colClasses)))
         ) stop("Matrix colnames and class names should match")
   class.levels<-levels(as.factor(colClasses))
   class.means<-sapply(class.levels, function(x){
     rowMeans(matrix[,names(colClasses)[colClasses == x]])
     })
   intra.var<-vector("numeric", length(class.levels))
   names(intra.var) <- class.levels
   inter.var.sum<-vector("numeric", length(class.levels))
   names(inter.var.sum) <- class.levels
   for (i in class.levels){
```

```
> op<-par(mfrow = c(1,1), pty = "s")
> par(mfrow = c(5,6), mar = c(1,1,1,1))
> n <- N
> for (i in 1:30){
   plot.new()
   axis(1, seq(0,1,0.5))
   axis(2, seq(0,1,0.5))
   title(paste("10^", n[i], sep = ""))
   lines(x = seq(0.01, 1, 0.01), y = PR.samples.all[[i]]$PR, col = "red")
   lines(x = seq(0.01, 1, 0.01), y = PR.samples.all.random[[i]]$PR, col = "blue")
   }
>   par(op)
```

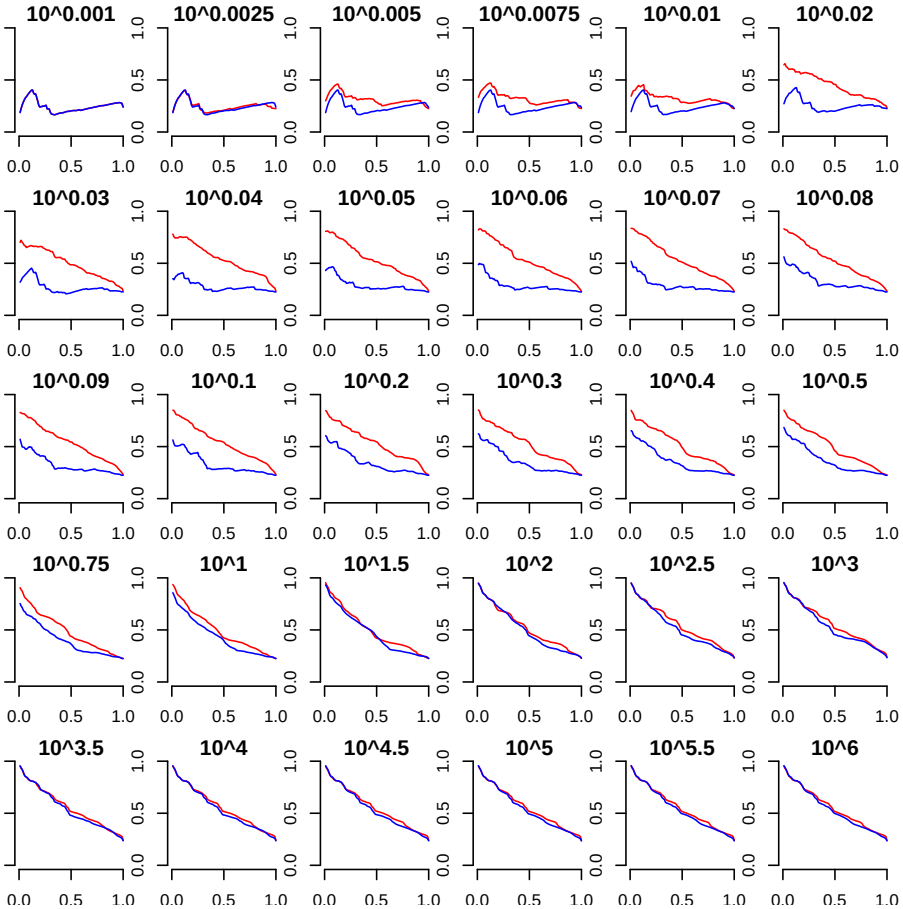

Figure 6: Precision-recall curves for aggregated data for pathways (red) and randomly constructed genesets (blue)

```
> plot(x = 10^(-N),
       y = unlist(lapply(
         PR.samples.all, function(x){x[["MAP"]]}
         )),
       log = "x",
       xlab = "Ternary threshold",
       ylab = "Mean average precision",
       ylim = c(0,1)
       , col = "red")
> points(x = 10^(-N),
       y = unlist(lapply(
         PR.samples.all.random, function(x){x[["MAP"]]}
         )), col = "blue")
> abline(h = unthresholded.MAP, col = "red")
```

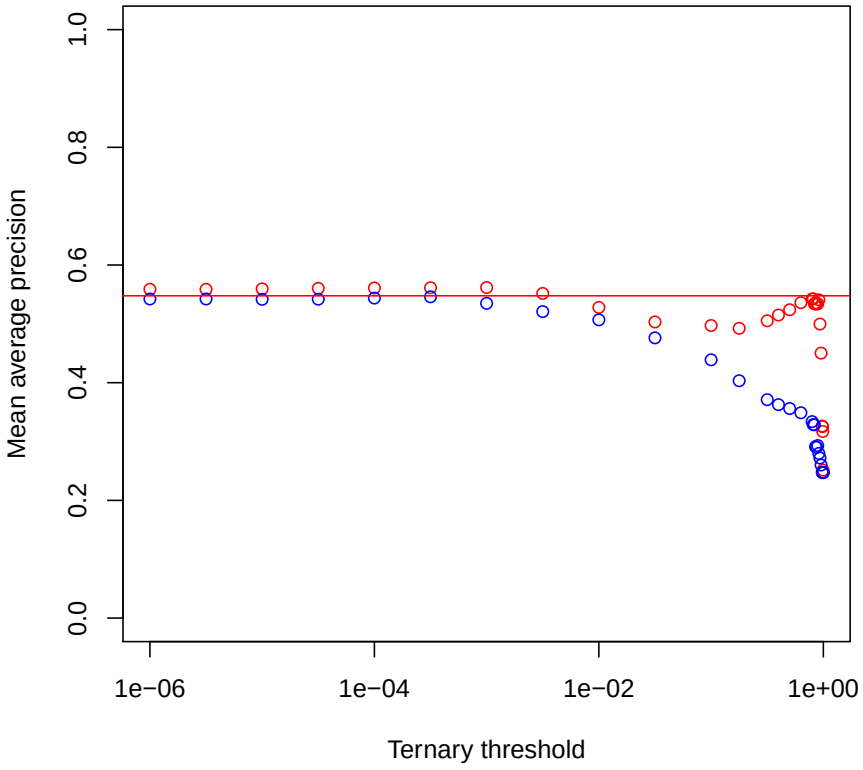

Figure 7: Plot of the mean average precision over the range of thresholds tested for the aggregated dataset (approach 3). Points indicates the mean average precision for the pathways (red) and randomly constructed genesets (blue). Solid red line indicates the mean average precision for the unthresholded fingerprint (red)

```
    grp.1 <- names(colClasses)[colClasses == i]

    intra.var[i] <- sum((dist(t(cbind(class.means[,i], matrix[,grp.1])),
                        method = method)[1:ncol(matrix[,grp.1])]
                )^2)
    inter.var <- vector("numeric", length(class.levels[class.levels != i]))
    names(inter.var) <- class.levels[class.levels != i]
    for (j in class.levels[class.levels != i]){
        inter.var[j] <- sum((dist(t(cbind(class.means[,j], matrix[,grp.1])),
                        method = method)[1:ncol(matrix[,grp.1])]
                )^2)
    }
    inter.var.sum[i] <- sum(inter.var)
    }
  ratio <- sum(intra.var)/sum(inter.var)
  return(ratio)
  }


> thresholds.varRatio.manhattan<-invisible(foreach (j = 1:30) %dopar% {
  n <- N[j]
  threshold<-10^(-n)
  full.threshold<-ternaryThreshold(brain.POE.matrix.naRm, threshold)
  varRatio(full.threshold, class, method = "manhattan")
  })
> unthresholded.varRatio.manhattan<-varRatio(brain.POE.matrix.naRm,
                                             class,
                                             method = "manhattan")
> thresholds.varRatio.euclidean<-invisible(foreach (j = 1:30) %dopar% {
  n <- N[j]
  threshold<-10^(-n)
  full.threshold<-ternaryThreshold(brain.POE.matrix.naRm, threshold)
  varRatio(full.threshold, class, method = "euclidean")
  })
> unthresholded.varRatio.euclidean<-varRatio(brain.POE.matrix.naRm,
                                             class,
                                             method = "euclidean")


> plot(x = 10^(-N),
     y = unlist(thresholds.varRatio.manhattan),
     log = "x",
     xlab = "Ternary threshold",
     ylab = "Intra/Inter cluster variance",
     ylim = c(0, max(c(1,max(unlist(thresholds.varRatio.manhattan)))))
     )
> abline(h = unthresholded.varRatio.manhattan, lty = 2)
```

```
> plot(x = 10^(-N),
      y = unlist(thresholds.varRatio.manhattan),
      log = "x",
      xlab = "Ternary threshold",
      ylab = "Intra/Inter cluster variance",
      ylim = c(0, 1.5)
      )
> abline(h = unthresholded.varRatio.manhattan, lty = 2)


> plot(x = 10^(-N),
      y = unlist(thresholds.varRatio.euclidean),
      log = "x",
      xlab = "Ternary threshold",
      ylab = "Intra/Inter cluster variance",
      ylim = c(0, max(c(1,max(unlist(thresholds.varRatio.euclidean)))))
      )
> abline(h = unthresholded.varRatio.euclidean,  lty = 2)


> plot(x = 10^(-N),
      y = unlist(thresholds.varRatio.euclidean),
      log = "x",
      xlab = "Ternary threshold",
      ylab = "Intra/Inter cluster variance",
      ylim = c(0, 1.5)
      )
> abline(h = unthresholded.varRatio.euclidean,  lty = 2)
```

7.1 K-fold cross validation

The data sets can be divided into K subsets of equal, or approximately equal, size. One of the subsets is omitted and mean pathway fingerprints calculated for each tissue from the remaining samples. Next, the samples in the omitted subset are assigned to the tissue with the closest mean tissue pathway fingerprint (by Euclidean or Manhattan distance) and these assignments compared to the known annotations, from which an an error rate is calculated. This is repeated, omitting each of the subsets in turn, to obtain a mean error rate. This cross validation procedure is performed 10 times for each threshold value to estimate the mean and standard deviation of the error rate. This proceedure is performed here for K = 5 and K = sample size (i.e. leave-one-out cross validation).

```
> crossValidation<-function(matrix, colClasses, K = 5, method = "manhattan")
   {
   # check values
   if (!(all.equal(colnames(matrix), names(colClasses)))) stop(
```

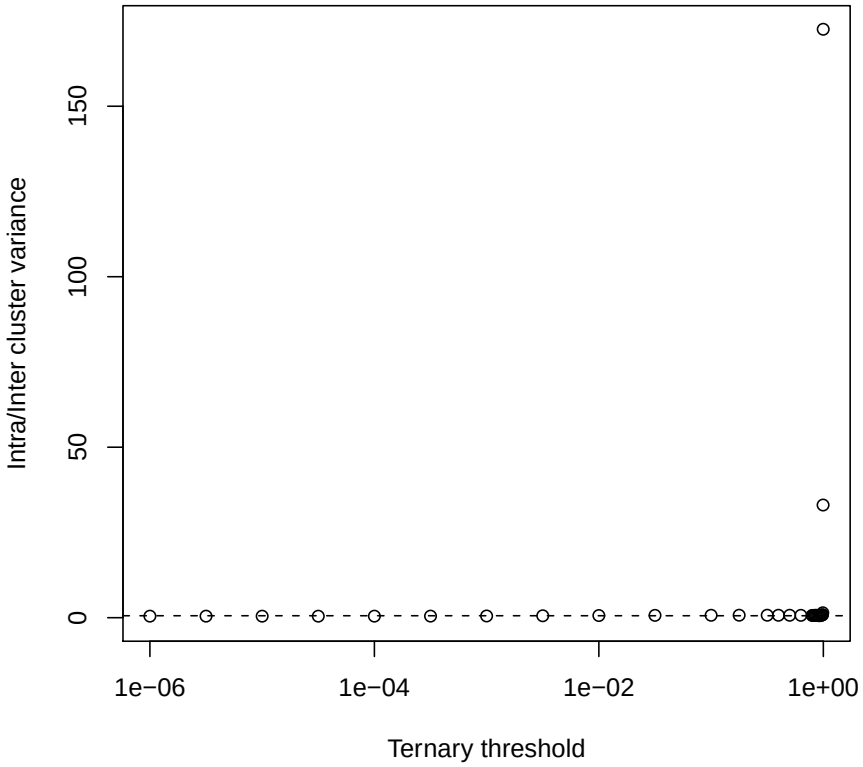

Figure 8: Plot of the intra-cluter vs inter-cluster variance ratio over the range of thresholds tested for the aggregated dataset. Variance is defined based on the Manhattan distance. Dashed line indicates the ratio for the unthresholded POE matrix

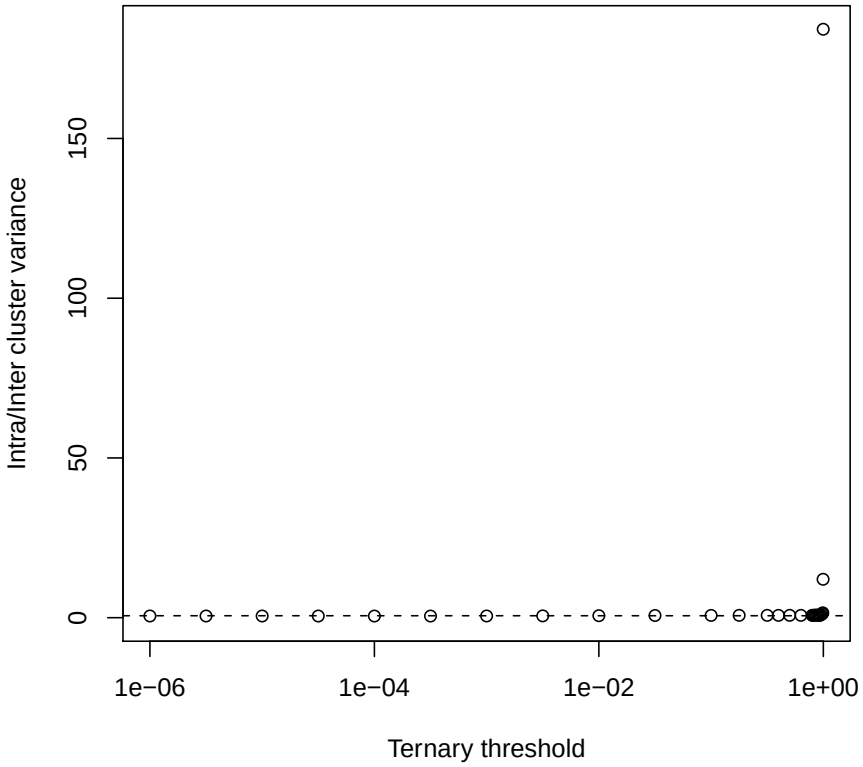

Figure 9: Plot of the intra-cluter vs inter-cluster variance ratio over the range of thresholds tested for the aggregated dataset. Variance is defined based on the Euclidean distance. Dashed line indicates the ratio for the unthresholded POE matrix

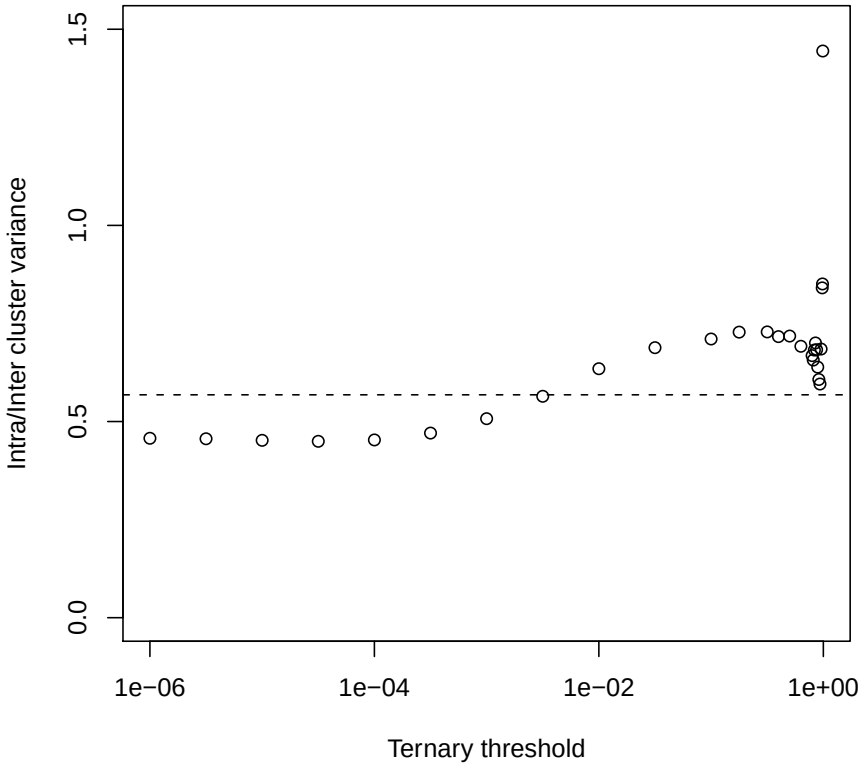

Figure 10: Plot of the intra-cluter vs inter-cluster variance ratio over the range of thresholds tested for the aggregated dataset. Variance is defined based on the Manhattan distance. Dashed line indicates the ratio for the unthresholded POE matrix - fixed y min/max

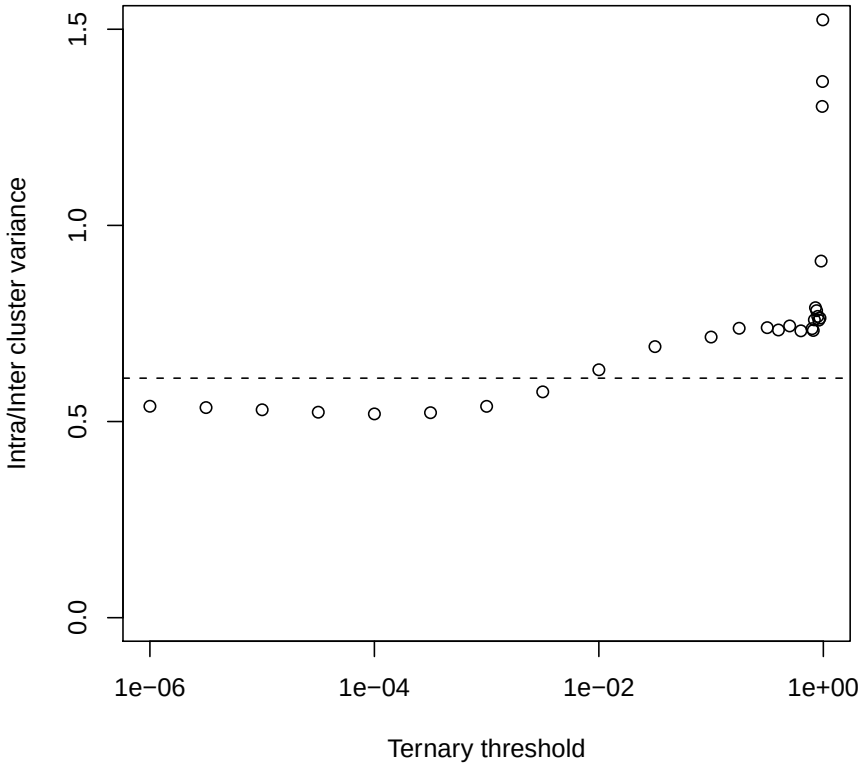

Figure 11: Plot of the intra-cluter vs inter-cluster variance ratio over the range of thresholds tested for the aggregated dataset. Variance is defined based on the Euclidean distance. Dashed line indicates the ratio for the unthresholded POE matrix - fixed y min/max

```
  "Check test matrix and class names shouldmatch")
if ((K > length(colClasses)) | (K <=1)) stop(
  "K outside allowable range")
# split into groups
n <- ncol(matrix)
K.o <- K
K <- round(K)
kvals <- unique(round(n/(1L:floor(n/2))))
temp <- abs(kvals - K)
  if (!any(temp == 0))
      K <- kvals[temp == min(temp)][1L]
  if (K != K.o)
      warning("K has been set to ", K)
  f <- ceiling(n/K)
  s <- sample(rep(1L:K, f), n)
  n.s <- table(s)
  ms <- max(s)
  CV <- vector("numeric", length(seq_len(ms)))
  for (i in seq_len(ms)) {
      j.out <- seq_len(n)[(s == i)]
      j.in <- seq_len(n)[(s != i)]
      matrix.out <- matrix[,j.out, drop = FALSE]
      matrix.in <- matrix[,j.in]
      colClasses.out <- colClasses[colnames(matrix.out)]
      colClasses.in <- colClasses[colnames(matrix.in)]
      # Insert function here
      class.levels.in<-levels(as.factor(colClasses.in))
      class.means.in<-sapply(class.levels.in, function(x){
        rowMeans(matrix.in[,names(colClasses.in)[colClasses.in == x]])
          })
      predictedTissue<-vector("character", ncol(matrix.out))
      for (j in 1:ncol(matrix.out)){
        if (method == "manhattan"){
          predictedTissue[j] <- names(which.min(colSums(abs(apply(class.means.in,
                        2,
                        function(x){x - matrix.out[,j, drop = FALSE]}
                        )))))
          }
      else if (method == "euclidean"){
          predictedTissue[j] <- names(which.min(colSums(abs(apply(class.means.in,
                        2,
                        function(x){(x - matrix.out[,j, drop = FALSE])^2}
                        )))))
        }
      }
      Actual <- as.factor(colClasses.out)
```

```
        Predicted<-factor(predictedTissue, levels = levels(Actual))
        m <- table(Actual = Actual, Predicted = Predicted)
        err <- (1 - sum(diag(m)) / sum(m))
        CV[i]<-err
    }
  # NaN occurs when there are no correct entries, i.e. error rate is 1
  CV[is.na(CV)] <- 1
  CV.av<-mean(CV)
  return(CV.av)
}

> # Manhattan distance
> fiveFold.manhattan<-invisible(foreach (j = 1:30) %dopar% {
  n <- N[j]
  threshold<-10^(-n)
  full.threshold<-ternaryThreshold(brain.POE.matrix.naRm, threshold)
  sapply(1:10, function(x){
    crossValidation(matrix = full.threshold,
    colClasses = class,
    K = 5,
    method = "manhattan")
      }
        )
  })
> unthresholded.fiveFold.manhattan<-sapply(1:10, function(x){
    crossValidation(matrix = brain.POE.matrix.naRm,
    colClasses = class,
    K = 5,
    method = "manhattan")
    }
   )
> # Euclidean distance
> fiveFold.euclidean<-invisible(foreach (j = 1:30) %dopar% {
  n <- N[j]
  threshold<-10^(-n)
  full.threshold<-ternaryThreshold(brain.POE.matrix.naRm, threshold)
  sapply(1:10, function(x){
    crossValidation(matrix = full.threshold,
    colClasses = class,
    K = 5,
    method = "euclidean")
      }
        )
  })
> unthresholded.fiveFold.euclidean<-sapply(1:10, function(x){
    crossValidation(matrix = brain.POE.matrix.naRm,
```

```
     colClasses = class,
     K = 5,
     method = "euclidean")
     }
    )
```

```
> library(gplots)
```

```
gdata: read.xls support for 'XLS' (Excel 97-2004)
gdata: files ENABLED.

gdata: read.xls support for 'XLSX' (Excel 2007+)
gdata: files ENABLED.
```

```
> plotCI(x = 10^(-N),
      y = sapply(fiveFold.manhattan, mean),
      uiw = sapply(fiveFold.manhattan, sd),
      log = "x",
      xlab = "Ternary threshold",
      ylab = "Mean error rate",
      ylim = c(0, 1),
      gap = 0,
      main = "Manhattan Distance"
      )
> abline(h = mean(unthresholded.fiveFold.manhattan), lty = 1)
> abline(h = mean(unthresholded.fiveFold.manhattan)+
   sd(unthresholded.fiveFold.manhattan), lty = 2)
> abline(h = mean(unthresholded.fiveFold.manhattan)-
   sd(unthresholded.fiveFold.manhattan), lty = 2)


> plotCI(x = 10^(-N),
      y = sapply(fiveFold.euclidean, mean),
      uiw = sapply(fiveFold.euclidean, sd),
      log = "x",
      xlab = "Ternary threshold",
      ylab = "Mean error rate",
      ylim = c(0, 1),
      gap = 0,
      main = "Euclidean Distance"
      )
> abline(h = mean(unthresholded.fiveFold.euclidean), lty = 1)
> abline(h = mean(unthresholded.fiveFold.euclidean)+
   sd(unthresholded.fiveFold.euclidean), lty = 2)
> abline(h = mean(unthresholded.fiveFold.euclidean)-
   sd(unthresholded.fiveFold.euclidean), lty = 2)
```

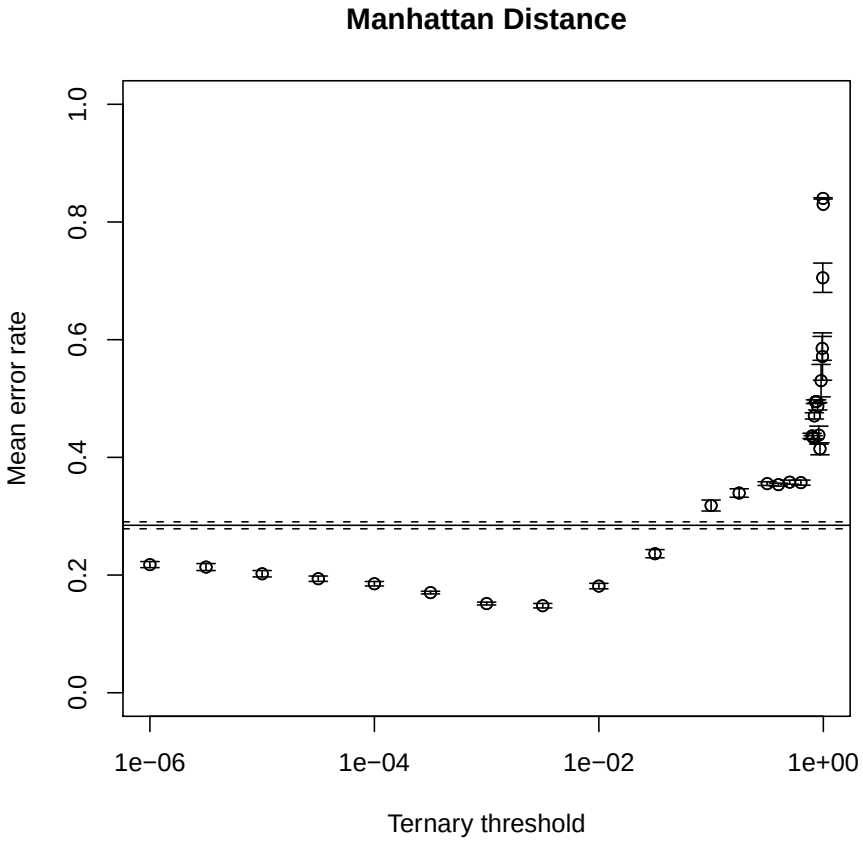

Figure 12: Plot of the average error rate based on 10 repeats of a 5-fold cross-validation over the range of thresholds tested for the aggregated dataset based on the Manhattan distance. Error bars indicate -/+ 1 std.dev. The black line indicates the ratio for the unthresholded POE matrix, dashed lines indicate -/+ 1 std.dev

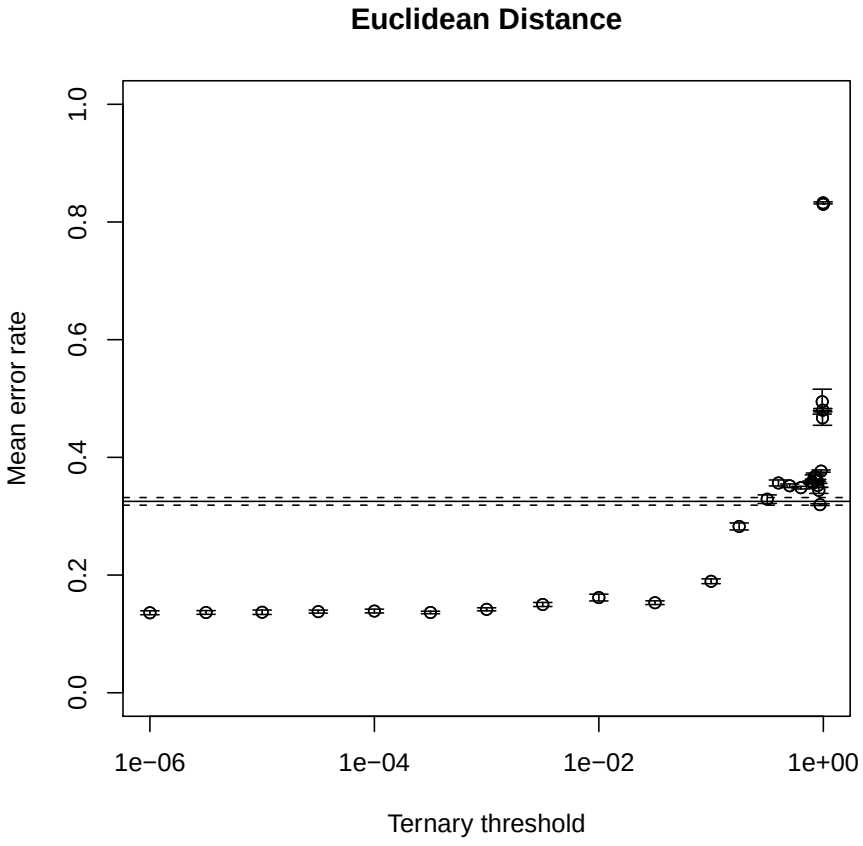

Figure 13: Plot of the average error rate based on 10 repeats of a 5-fold cross-validation over the range of thresholds tested for the aggregated dataset based on the Euclidian distance. Error bars indicate -/+ 1 std.dev. The black line indicates the ratio for the unthresholded POE matrix, dashed lines indicate -/+ 1 std.dev

```
> # Manhattan distance
> LOO.manhattan<-invisible(foreach (j = 1:30) %dopar% {
   n <- N[j]
   threshold<-10^(-n)
   full.threshold<-ternaryThreshold(brain.POE.matrix.naRm, threshold)
   crossValidation(matrix = full.threshold,
     colClasses = class,
     K = length(class),
     method = "manhattan")
   })
> unthresholded.LOO.manhattan<-
     crossValidation(matrix = brain.POE.matrix.naRm,
     colClasses = class,
     K = length(class),
     method = "manhattan")
> # Euclidean distance
> LOO.euclidean<-invisible(foreach (j = 1:30) %dopar% {
   n <- N[j]
   threshold<-10^(-n)
   full.threshold<-ternaryThreshold(brain.POE.matrix.naRm, threshold)
   crossValidation(matrix = full.threshold,
     colClasses = class,
     K = length(class),
     method = "euclidean")
   })
> unthresholded.LOO.euclidean<-
     crossValidation(matrix = brain.POE.matrix.naRm,
     colClasses = class,
     K = length(class),
     method = "euclidean")


> library(gplots)
> plotCI(x = 10^(-N),
       y = sapply(LOO.manhattan, mean),
       uiw = sapply(LOO.manhattan, sd),
       log = "x",
       xlab = "Ternary threshold",
       ylab = "Mean error rate",
       ylim = c(0, 1),
       gap = 0,
       main = "Manhattan Distance")
> abline(h = unthresholded.LOO.manhattan, lty = 1)


> plotCI(x = 10^(-N),
       y = sapply(LOO.euclidean, mean),
```

```
      uiw = sapply(LOO.euclidean, sd),
      log = "x",
      xlab = "Ternary threshold",
      ylab = "Mean error rate",
      ylim = c(0, 1),
      gap = 0,
      main = "Euclidean Distance")
> abline(h = unthresholded.LOO.euclidean, lty = 1)
```

Figure 14: Plot of the error rate based on a leave-one-out cross-validation over the range of thresholds tested for the aggregated dataset based on the Manhattan distance. The black line indicates the ratio for the unthresholded POE matrix, dashed lines indicate -/+ 1 std.dev

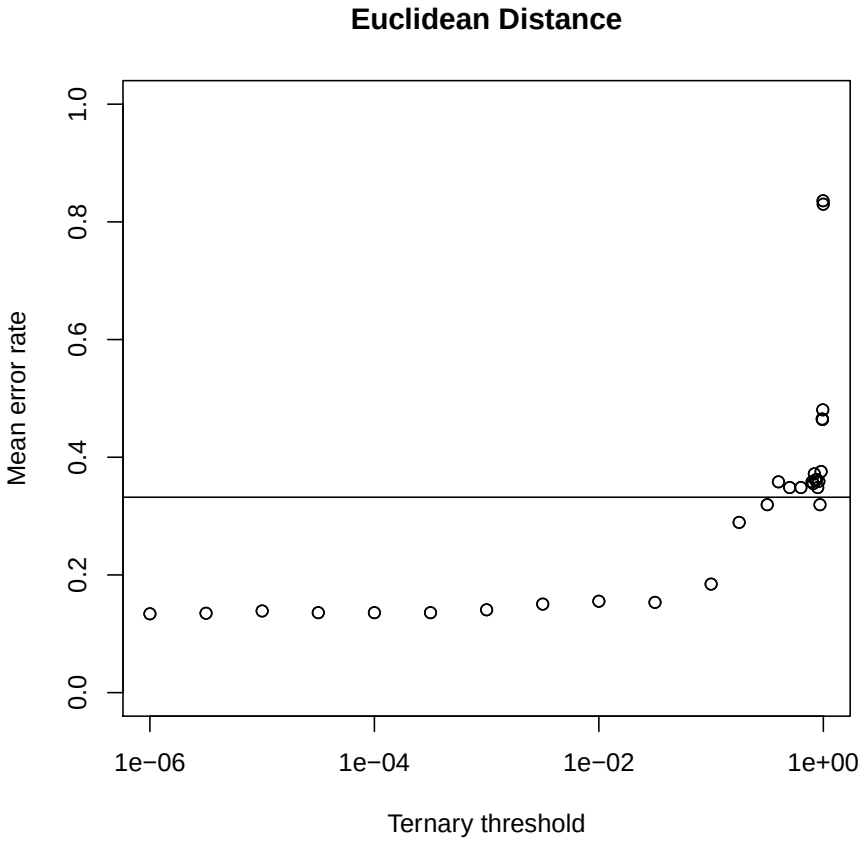

Figure 15: Plot of the error rate based on a leave-one-out cross-validation over the range of thresholds tested for the aggregated dataset based on the Euclidian distance. The black line indicates the ratio for the unthresholded POE matrix